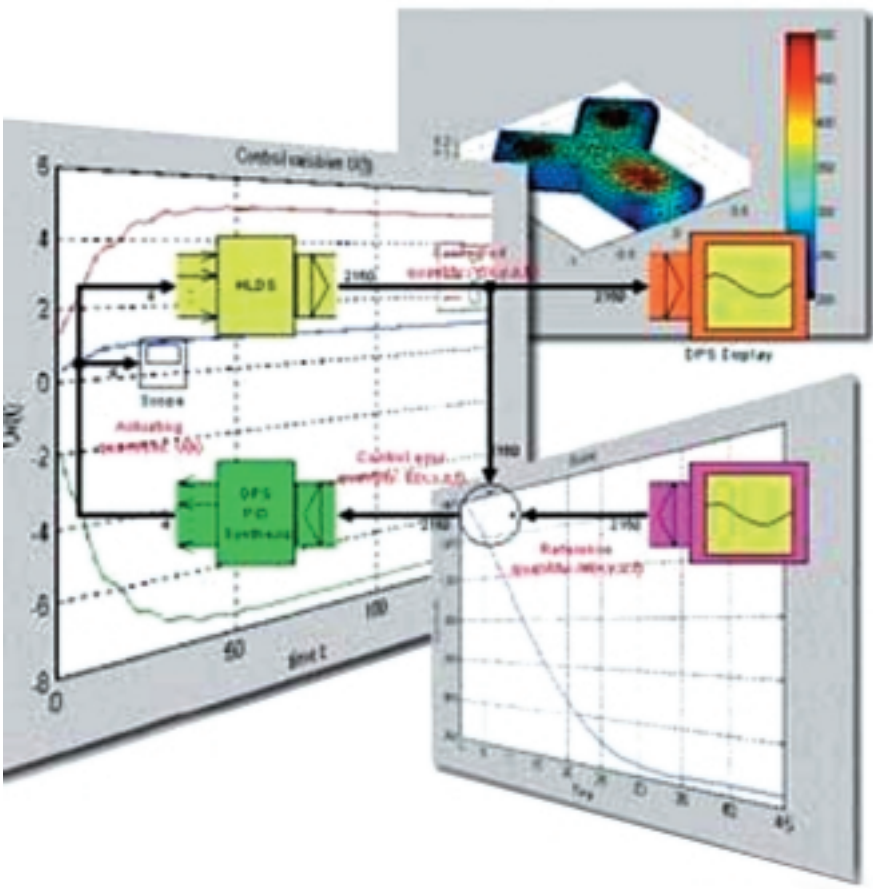
Codiseño de un sistema de control en red para regular la velocidad de un motor DC

Diego Martínez,*
Fernando I. Arévalo,*
Carlos F. Giraldo,*
Apolinar González**



Resumen

En este trabajo se presenta el codiseño del algoritmo de control y de la plataforma de implementación de un sistema en red que regula la velocidad de un motor DC, para lo cual, desde el análisis del algoritmo de control, se consideran las características de la plataforma de implementación. Los resultados de simulación y experimentales permiten validar el diseño realizado. Las características de los niveles que conforman el sistema son consideradas en su totalidad durante las fases de diseño; es decir, en una misma representación se analizan los retardos, el flujo de datos y la funcionalidad de los componentes del sistema, permitiendo disminuir la diferencia entre los resultados experimentales y los resultados de simulación. El procedimiento de diseño utilizado es el resultado de integrar varias técnicas y metodologías para especificar y diseñar sistemas de tiempo real, lo cual permite utilizar las

* Departamento de Automática y Electrónica, Grupo de Investigación Sistemas de Telemando y Control distribuido.Universidad Autónoma de Occidente. Santiago de Cali, Colombia. dmartinez, fiarevalo, cfgiraldo@uao.edu.co;

** Facultad de Ingeniería Mecánica y Eléctrica, Universidad de Colima. Colima, México. apogon@ucol.mx

Fecha de recepción: 02/16/05, Fecha de aprobación: 04/20/05

ventajas presentadas en estas técnicas, tales como el codiseño hardware-software.

Palabras clave: Sistemas de control distribuido, Tiempo real, Sistemas embebidos, Codiseño Hardware-Software, Redes de petri coloreadas.

Abstract

This work shows the codesign of the control and the implementation architecture in a scheme of networked control systems that regulates the speed of a DC motor; since the control algorithm analysis the characteristics of the implementation architecture are considered. The experimental results and of simulation allow to validate the design made. The characteristics of the levels that conform this system are considered in their totality during the phases of designing; it means, at he same representation are analyzed the retardations, the data flow and the system components functionality, allowing to diminish the different between the experimental results and the results of simulation.

The design procedure used is the result of integrating several techniques and methodologies to specify and to design real time systems, which allow to use the advantages displayed in these techniques, such as the codesign hardware - software.

Key words: Distributed control systems, Real time, Embedded systems, Codesign Hardware-Software, Colored petri nets.

1. Introducción

Debido al aumento en la complejidad de los sistemas de control gran parte de las actividades se deben distribuir entre diferentes nodos, los cuales se comunican por medio de redes de comunicación.

Adicionalmente la implementación de sistemas distribuidos permite disminuir el impacto producido por las fallas en un componente del sistema y facilita las actividades de diagnóstico y mantenimiento del mismo.

De otro lado, al diseñar sistemas de control en red con dinámicas muy exigentes no siempre se obtiene una buena correspondencia entre los resultados experimentales y los de simulación, esto es debido al uso de modelos imprecisos para analizar y diseñar estos sistemas, métodos de validación poco elaborados y plataformas que no soportan los modelos empleados.

El análisis y diseño de sistemas de control en red es una disciplina relativamente nueva, por lo que en la literatura existen muy pocas publicaciones que hacen referencia a técnicas de modelado y simulación.[1,5,12] Actualmente esta disciplina presenta una serie de desafíos. Entre ellos se encuentran el desarrollo de herramientas, métodos y modelos para soportar las etapas arquitecturales del diseño; facilitar la comunicación entre los diseñadores de diferentes disciplinas; y realizar validaciones adecuadas.

Sin embargo, durante los últimos años los investigadores han mostrado un gran interés por desarrollar nuevos métodos de especificación.[2,3,4]

En este artículo se presenta el codiseño de un sistema de control en red que regula la velocidad de un motor DC, para lo cual se ha realizado una mejora al procedimiento de diseño de sistemas de control

distribuido de tiempo real presentado que permite resolver algunas de las dificultades presentes en el diseño de estos sistemas,[1,5] debido a que introduce en los análisis, las características del planificador de tareas y del protocolo de comunicaciones. Adicionalmente permite analizar el comportamiento, flujo de datos y evolución temporal en todos los componentes del sistema, todo en un mismo modelo a partir de redes de Petri Coloreadas (CPN). En este caso se emplean componentes para las diferentes fases de diseño, con el fin de modelar, analizar el comportamiento y validar el diseño de estos sistemas, de tal forma que los resultados obtenidos cumplan con las especificaciones propuestas en los sistemas de control.

El artículo está organizado de la siguiente forma. En la sección 2 se presenta el procedimiento de diseño utilizado. En la sección 3 el diseño de un sistema de control en red que regula la velocidad de un motor DC y en la sección 4 se presentan las conclusiones y el trabajo futuro.

2. Procedimiento de diseño de sistemas de control en red

La arquitectura propuesta para implementar el sistema de control se presenta en la Figura 1, en ella se pueden observar tres niveles:

1. *Interface de usuario*. Las aplicaciones en este nivel se implementan en una computadora personal (PC). Dependiendo de los requerimientos presentes, para el diseño de componentes en este nivel se puede requerir la utilización de lenguajes y sistemas operativos especiales para aplicaciones de tiempo real.
2. *Red de comunicaciones*. Para interconectar los nodos del sistema se seleccionó el protocolo CAN, con el cual se logran retardos variables en la transmisión pero acotados, y se maneja una información consistente entre todos los nodos.
3. *Módulos de bajo nivel*. En este campo se implementan las funciones de medición, y cálculo de acción de control y actuación. Para su diseño se utilizan microprocesadores y FPGA, y se integran conceptos de la metodología de codiseño hardware-software.

2.1. Procedimiento para diseñar sistemas de control en red con restricciones de tiempo real

El procedimiento de diseño utilizado en este trabajo se explica en la Figura 2 y es una versión mejorada del presentado en el análisis y diseño de sistema de control en red.[1,5,12]

Las etapas del procedimiento son:

1. *Diseño arquitectural*. En esta fase se desarrolla un modelo arquitectural del sistema que cumple con las especificaciones planteadas, para lo cual se utilizan los componentes propuestos. Durante este proceso no se definen las plataformas sobre las cuales será implementado cada uno de los componentes.
2. *Asignación de componentes*. En esta fase se definen las plataformas sobre las cuales será implementado cada uno de los componentes.
3. *Diseño del modelo en CPN*. Durante esta etapa, utilizando las representaciones básicas de cada componente en CPN, se traslada el modelo arquitectural del sistema a un modelo en CPN, que permite realizar una validación funcional del sistema.
4. *Diseño y síntesis de componentes*. Utilizando los lenguajes y

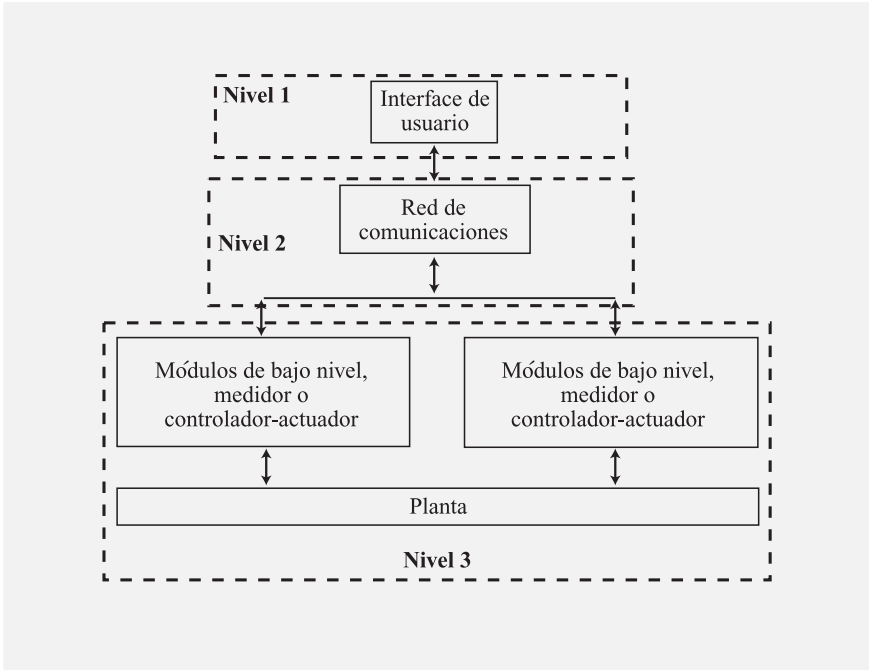


Figura 1. Arquitectura del sistema de control distribuido.

plataformas seleccionados se realiza el diseño y síntesis de cada uno de los componentes del sistema.

5. *Validación del sistema*. Durante esta fase se realiza la medición de los tiempos de cómputo de cada una de las actividades y se caracteriza el modelo obtenido en CPN, con el cual se realiza la validación dinámica del sistema considerando los aspectos funcionales, de tiempo real y el procesamiento de la información; adicionalemente se evalúa la función de costo del sistema.

En función de los resultados obtenidos de la validación es posible realizar:

a) Si se detecta un problema funcional se regresa a la etapa 3.

b) Si la asignación de componentes no corresponde a aquella que minimiza la función de costo propuesta, se debe regresar a la etapa 2, o a la etapa 1.

c) Si los resultados son satisfactorios, se realiza la implementación e integración del sistema.

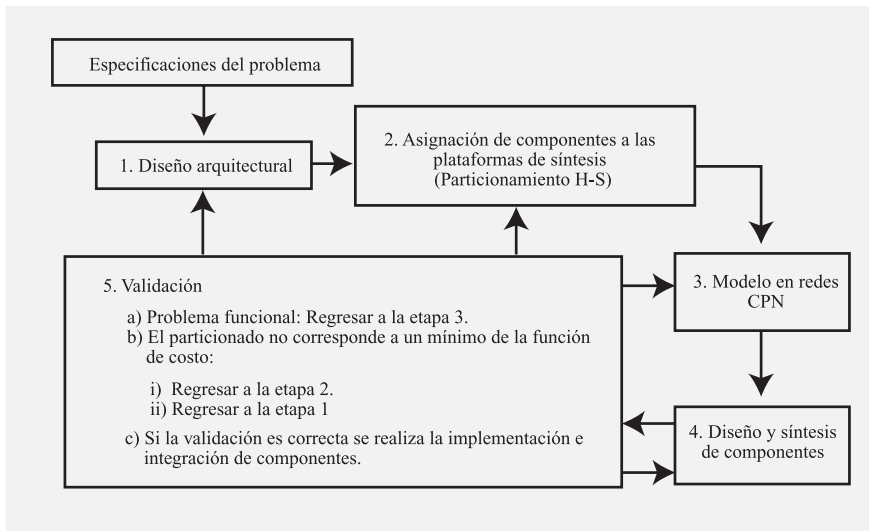


Figura 2. Etapas del procedimiento de diseño

2.2. Selección de lenguajes de especificación

Con el fin de lograr un adecuado diseño e implementación se seleccionaron lenguajes que soportaran los modelos propuestos. Para el desarrollo del modelo arquitectural se seleccionaron los componentes descritos en la metodología HRT-HOOD. Sin embargo, para realizar la descripción arquitectural de sistemas distribuidos, fue necesario adicionar tres componentes a los descritos en HRT-HOOD; en el pro-

cedimiento de diseño de sistemas de control[1,5] se presenta una descripción completa de los componentes propuestos para el diseño.

En este trabajo se seleccionó y diseñó un planificador basado en prioridades estáticas con expropiación, el cual coordina la ejecución de los procesos implementados en software en los módulos de bajo nivel, debido a que presenta un buen comportamiento en presencia de sobrecarga.

Para realizar la descripción del comportamiento y el análisis del sistema se seleccionaron las CPN, que poseen una semántica bien definida y permiten en una misma representación modelar jerarquías; analizar tiempos, flujo y procesamiento de datos, y estudiar el comportamiento de los componentes del sistema (estados, acciones, concurrencia, etc.).

La validación de los modelos en CPN se realizó en forma dinámica utilizando la herramienta CPN tools.

Los criterios de selección de los componentes y lenguajes, y la representación de los componentes en CPN se detalla en el procedimiento de diseño de sistemas de control.[5]

3. Codiseño de un sistema de control en red para regular la velocidad de un motor DC

El diseño de un sistema de control posee dos etapas: en la primera se realiza una selección del algoritmo de control, se calculan los coeficientes del mismo, y se seleccionan el período de muestro y los retardos máximos admitidos en el lazo de realimentación. Durante la segunda etapa se realiza el diseño e implementación de la aplicación.

Debido a la gran dependencia que existe entre estas etapas, en los últimos años los investigadores han mostrado un gran interés por el desarrollo de procedimientos de diseño conjunto del algoritmo de control y de la plataforma de implementación, para lo cual se han desarrollado métodos de análisis de estabilidad de sistemas de control en red y algoritmos de control que compensan los efectos producidos por las plataformas de implementación sobre la dinámica de los sistemas de control.[6,7,8]

3.1 Diseño del algoritmo de control y análisis dinámico del sistema

Para regular la velocidad del motor se obtuvo un modelo lineal de primer orden que representa la dinámica del sistema a controlar, y se diseñó un algoritmo PI de la forma:

$$U_{(z)} = \frac{k_p z + (k_i T - k_p)}{z - 1} \varepsilon_{(z)} \quad (1)$$

Donde «T» representa el período de muestreo. La Figura 3 presenta la respuesta deseada del sistema de control.

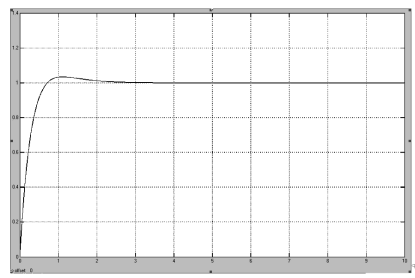


Figura 3. Respuesta deseada del sistema de control.

Para realizar el análisis dinámico del sistema de control se utilizaron las transformadas de Pade y Bilineal, a partir de lo cual se obtuvo un modelo del sistema en lazo cerrado en términos del período de muestreo y del retardo en el lazo de realimentación, con el cual se analizó la dinámica del sistema en lazo cerrado para diferentes retardos y períodos de muestreo y se generó la región de estabilidad para retar-

dos menores o iguales al período de muestreo. La Figura 4 presenta los resultados obtenidos en simulación.

En la Figura 4a, para un T = 100 ms y un retardo (t) del 40% del valor de T, se puede apreciar un desempeño oscilatorio de la respuesta de sistema de control en el cual se implementó la ecuación (1).

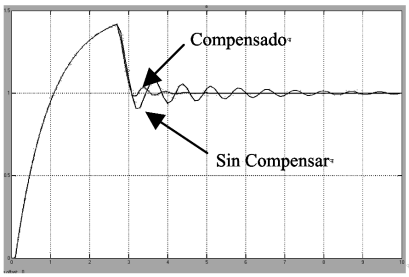


Figura 4a. Sistema de control de velocidad con T = 100 ms y retardo del 40% de T, algoritmo PI con compensador y sin él.

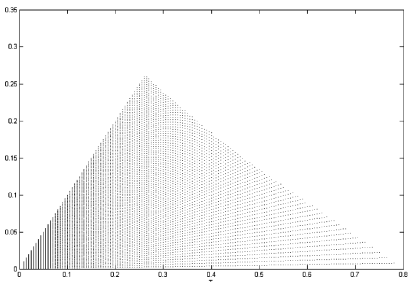


Figura 4b. Región de estabilidad del sistema de control en red para regular la velocidad del motor DC.

Figura 4. Análisis del comportamiento dinámico del sistema de control de velocidad.

Con el fin de compensar el efecto del retardo presente en el lazo de realimentación, el cual en nuestro caso es constante, se implementó un algoritmo PI que utiliza un módulo de compensación.[8]

$$E_{\Delta(z)} = \frac{Tz}{(T-\Delta)z+\Delta}\varepsilon_{(z)} \qquad (2)$$

Los resultados de simulación presentados en la Figura 4a muestran que la respuesta del algoritmo compensado se aproxima mucho más a las especificaciones deseadas del sistema. De esta forma se ha desarrollado un algoritmo de control que considera las características de la plataforma tecnológica sobre la cual será implementado.

A continuación se presentan las especificaciones con las cuales se diseñó el sistema de control de velocidad y los resultados de las etapas del diseño.

3.2 Especificaciones

Las especificaciones funcionales y temporales del sistema de control de velocidad son:

• Especificaciones funcionales

- *Funciones del módulo medidor.* Este módulo realiza el sensado de la velocidad y el acondicionamiento de esta señal. Sus funciones son:
- Medir y filtrar la velocidad del motor, y transmitir este dato a través de la red de comunicaciones CAN.
- *Funciones del módulo controlador.* Este módulo calcula la acción de control y actúa sobre el sistema. Sus funciones son:
  - Calcular la acción de control y enviar ésta al actuador.
  - Recibir la referencia y la medida de velocidad a través de la red de comunicaciones CAN.
  - Visualizar la referencia y la velocidad localmente en un LCD.
  - Transmitir la información de la acción de control a la interface de usuario a través de la red de comunicaciones CAN.
- *Funciones del módulo PC*. En este módulo se implementa la interface de usuario. Sus funciones son:

- Recibir los datos de la medición de nivel y la acción de control transmitidos a través de la red CAN, y visualizarlos.
- Transmitir a través de la red CAN las modificaciones que se realicen a la referencia.

- **Especificaciones temporales**

De acuerdo con los resultados obtenidos al analizar la dinámica del sistema, se realizó la siguiente selección para el período de muestro y retardo máximo admitido para toda la tarea de regulación, la cual involucra: medición, filtrado, transmisión y recepción de información, cálculo de acción de control y acción sobre el sistema.

- El período de muestreo de la velocidad es de 35 ms.
- El retardo máximo para la tarea de control es de un período de muestreo.

3.3 Etapas del diseño de la plataforma de implementación del sistema de control

Una vez finalizada la etapa en la que se encuentran los parámetros relacionados con el control, se continúa con el diseño e implementación de la aplicación.

En esta sección se presentan los resultados de las etapas que se realizaron para diseñar y aplicar el sistema de control en red que regula la velocidad del motor, mostrado en la Figura 5.

3.3.1 Diseño arquitectural del sistema de control

El nivel inicial del diseño arquitectural del sistema que regula la velocidad del motor DC se presenta en la Figura 6. En él se pueden apreciar tres nodos: medidor, controlador y PC. Estos nodos corresponden con la arquitectura del sistema presentada en la Figura 1.

3.3.2 Modelo del sistema en redes de Petri Coloreadas

La Figura 7 muestra el nivel de mayor jerarquía del modelo en redes de Petri Coloreadas del sistema de control de velocidad. En la Figura 8 se presenta el modelo del planificador de tareas basado en prioridades estáticas con expropiación.

Este modelo permite describir completamente la solución que se propuso para cumplir con las especificaciones propuestas, incluyendo todos los aspectos relacionados con su implementación.

La validación dinámica de esta representación permite corroborar

Figura 5. Planta de control de velocidad de un motor DC.

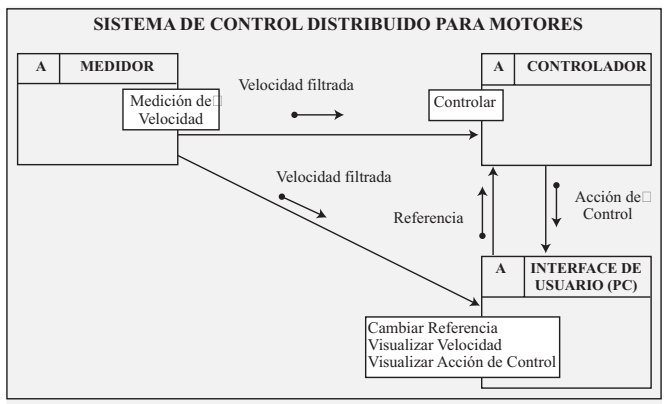


Figura 6. Nivel inicial del diseño arquitectural del sistema de control de velocidad.

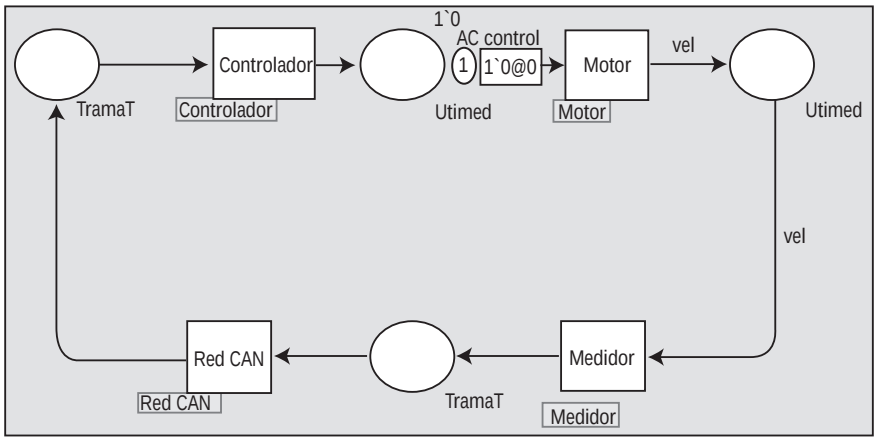


Figura 7. Modelo en CPN del sistema de control de velocidad.

el cumplimiento de las especificaciones del sistema considerando el comportamiento de todos los componentes utilizados en la implementación. La Tabla 1 presenta los parámetros temporales de las tareas del proceso de regulación durante dos ciclos de ejecución, realizados en CPN Tools, de los cuales se puede observar el cumplimiento de los plazos temporales impuestos por el sistema.

Tabla 1. Análisis del cumplimiento de los plazos en el modelo del sistema de control realizado en CPN.

Tarea	Parámetros temporales Ciclo 1			Parámetros temporales Ciclo 2		
	Inicio	Final	Cómputo	Inicio	Final	Cómputo
Medidor	35	35.71	0.71	70	70.71	0.71
Red CAN	35.71	36.13	0.42	70.71	71.13	0.42
Controlador	36.13	53.13	17	71.13	88.13	17

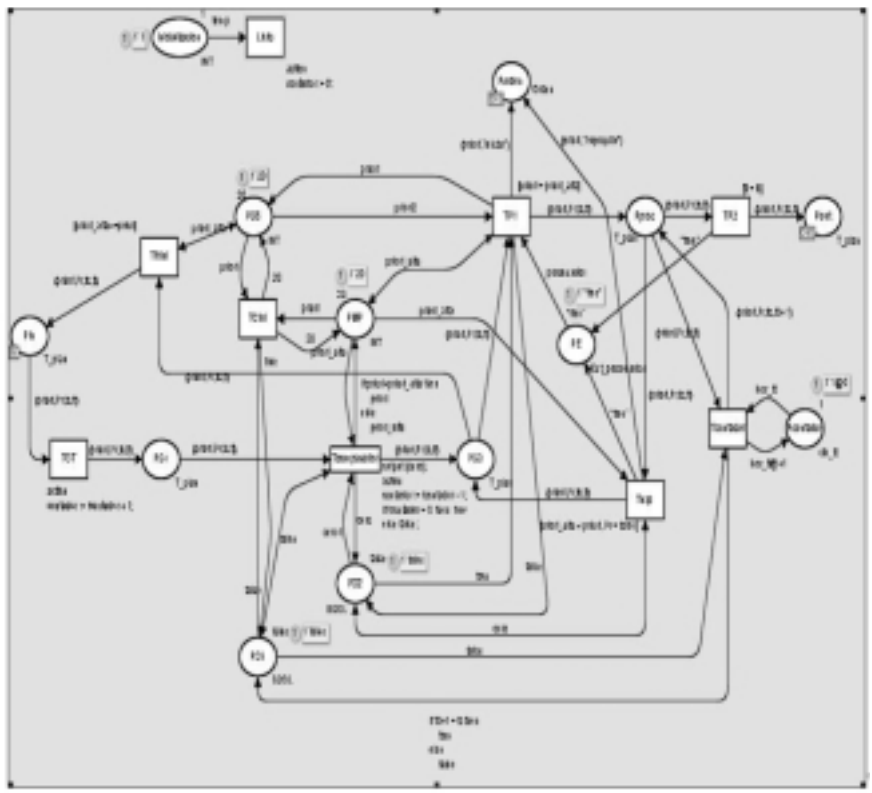


Figura 8. Modelo en CPN del planificador de tareas basado en prioridades estáticas.

Los tiempos presentados en la Tabla 1 están dados en milisegundos.

La Figura 9 presenta los resultados obtenidos de las validaciones dinámicas del modelo en CPN, tanto de la velocidad como de la acción de control.

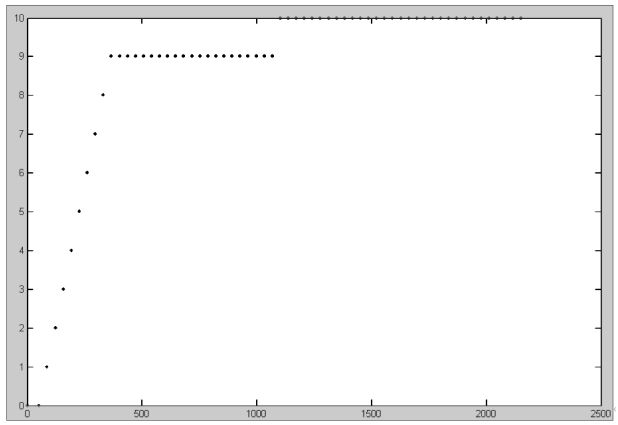


Figura 9a. Respuesta de la velocidad del motor analizado en CPN.

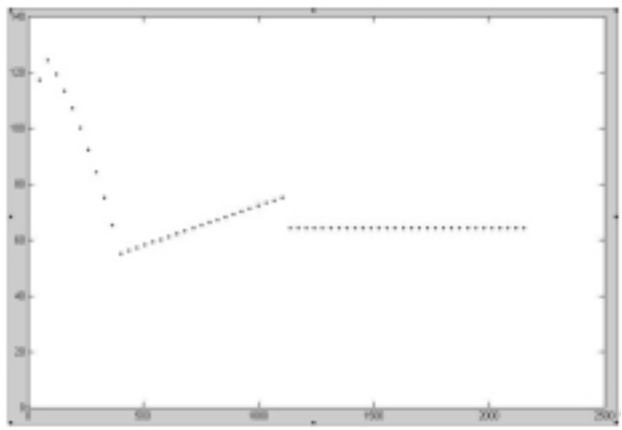


Figura 9b. Acción de control generada en el modelo del sistema realizado en CPN.

Figura 9. Resultados obtenidos de las validaciones dinámicas del modelo en CPN.

Los resultados presentados en la Figura 9 corresponden con los resultados de simulación y experimentales del sistema.

3.3.3 Implementación del sistema de control

En la selección de las plataformas de implementación intervienen factores como recursos, costo, disponibilidad, conocimiento, etc. Se propone un desarrollo total en FPGA utilizando módulos IP de procesadores para la implementación del software,[1] lo cual ofrece una gran flexibilidad. En este trabajo la aplicación de los componentes software se realizó sobre el microcontrolador 89C52 y los componentes hardware se desarrollaron sobre la FPGA Flex 10K70. El nodo controlador se aplicó completamente en software, para lo cual se diseñó un planificador de tareas con prioridades estáticas con expropiación. En el diseño del módulo medidor se aplicó la técnica de Codiseño Hardware-Software, con lo cual se logró obtener una adecuada implementación que cumple con las especificaciones propuestas.

La Figura 10 muestra los resultados de la simulación del componente que maneja la comunicación a través de la red CAN.

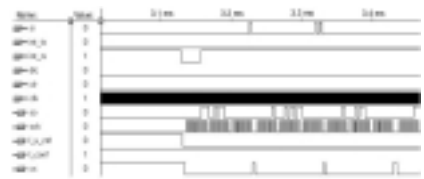


Figura 10. Resultados de la simulación del componente que maneja la comunicación a través de la red CAN.

La aplicación del módulo PC se realizó en una computadora personal, los componentes se desarrollaron en Visual Basic y fueron implementados sobre el sistema operativo Windows.

3.3.4 Codiseño del módulo medidor

Los parámetros considerados para evaluar el costo de realizar una asignación de componentes en hardware y software fueron los siguientes:

- *Ap,* cantidad de área requerida para la implementación de componentes en hardware.
- *Tp,* latencia de cómputo.
- *Mp,* cantidad de memoria requerida para la implementación de componentes en software.

En el diseño de un nodo medidor[12] se presenta la función de costo utilizada para realizar el Codiseño Hardware-Software de este módulo, la cual tiene la forma:

$$F(P)=\sum_{0}^{i} K_i \frac{C_i(P)}{C_i}+\sum_{0}^{i} K_{Ci} F_C[C_i, C_i(P)]$$

Donde *«Ci»* es la restricción de diseño aplicado al i-ésimo parámetro, *«Ci(P)»* es la solución obtenida de una nueva partición *«P»* que es usada como parámetro de normalización, *«Kci»* es el peso del factor para los términos de corrección y *«Fc»* es la función de corrección.

La función de corrección utilizada en este trabajo es una función de penalidad, que no contribuye a la función de costo cuando la solución está dentro del espacio permitido de búsqueda. La expresión de la función de penalidad es:

$$F_C(C_i, C_i(P))=r^2[C_i, C_i(P)]$$

Donde la función $r[C_i,C_i(P)]$ corresponde a:

$$r(C_i, C_i(P))=máx\left\{0, \frac{[C_i(P)-C_i]}{C_i}\right\}$$

El diseño arquitectural del nodo medidor se presenta en la Figura 11. En ella se puede apreciar que este nodo posee tres componentes: Lectura, Filtrado y Driver de red (Controlador CAN).

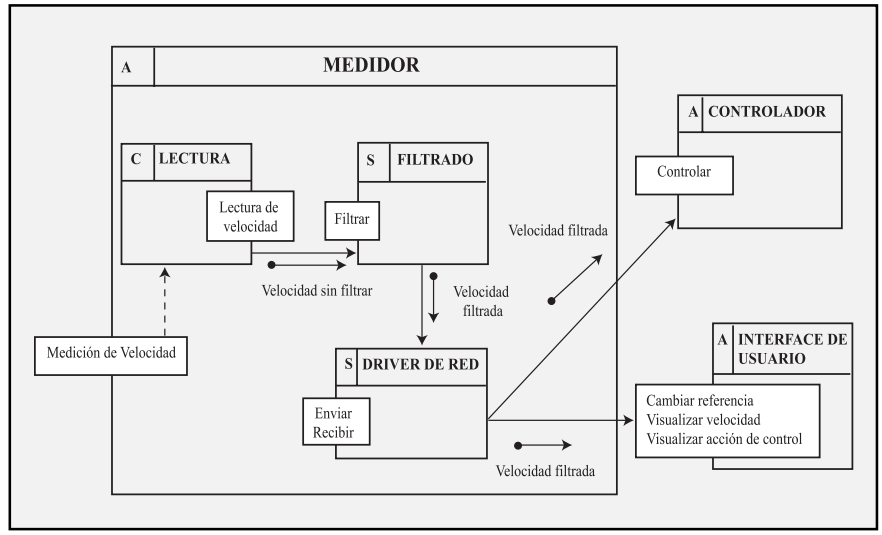


Figura 11. Diseño arquitectural del nodo medidor.

Los parámetros utilizados para realizar el análisis de costo fueron:

- Restricciones: Área = 913 Celdas Lógicas, Tiempo de cómputo = 2894 µs y *Memoria* = 798 Bytes.
- Coeficientes de las restricciones y de la función de corrección: K_A =0.3; K_T =0.4; K_M =0.3; K_{CA} = K_{CT} = K_{CM} =150.

Con estos parámetros, cada uno de los módulos del medidor fue descrito en VHDL y en código ensamblador. Las estimaciones de los parámetros de área requerida en hardware (*Ap*), cantidad de memoria requerida (*Mp*) y tiempo de ejecución (*Tp*) fueron realizadas analizando directamente las posibles particiones que se tendrían como espacio de diseño. El resultado de las estimaciones se presenta en la Tabla 2.

En este trabajo para encontrar la mejor partición se utilizaron los algoritmos de *Fiduccia-Mattheyses y Recosido Simulado,* con los cuales se obtuvo el mismo resultado. Definidos los valores de los parámetros de implementación y la función de costo, se puede iniciar con el particionado.

Tabla 2. Datos estimados del espacio de soluciones al diseño del nodo medidor.

	Área (Celdas lógicas)	Tiempo de cómputo (µs)	Memoria (Bytes)
Lectura (H) - Filtrado (H) - Driver (H)	913	736.24	0
Lectura (H) - Filtrado (H) - Driver (S)	146	2836.205	639
Lectura (H) - Filtrado (S) - Driver (H)	855	768.48	234
Lectura (H) - Filtrado (S) - Driver (S)	70	2868.48	719
Lectura (S) - Filtrado (H) - Driver (H)	855	761.725	228
Lectura (S) - Filtrado (H) - Driver (S)	70	2861.725	713
Lectura (S) - Filtrado (S) - Driver (H)	785	794	313
Lectura (S) - Filtrado (S) - Driver (S)	0	2894	798

Se empleó la siguiente abreviación para cada tarea:

- Lectura (L)
- Filtrado (F)
- Controlador CAN (D_C)

Partición inicial: Todo en software [(L,F,D_C),ϕ]

Costo de la Partición inicial: 0.700

En la primera iteración tenemos:

Tabla 3. Resultados de la primera iteración del algoritmo *Fiduccia-Mattheyses* aplicado al diseño del nodo medidor.

Tarea movida	Partición resultante	Costo de la partición resultante	¿Se mejora el costo?
L	[(F,D_C),L]	0.6898	Sí
F	[(L,D_C),F]	0.6866	Sí
D_C	[(L,F),D_C]	0.4854	Sí

Se puede observar de la Tabla 3 que el movimiento de la tarea *D_C* genera la mayor disminución del costo, por esta razón se deja quieta la tarea *D_C* y se continúa con la búsqueda de otro movimiento con las demás tareas que generen un menor costo.

Nueva partición de inicio: [(L,F),D_C]

Costo de la partición inicial: 0.4854

En la segunda iteración tenemos:

Tabla 4. Resultados de la segunda iteración del algoritmo *Fiduccia-Mattheyses* aplicado al diseño del nodo medidor.

Tarea movida	Partición resultante	Costo de la partición resultante	¿Se mejora el costo?
L	[F,(L,D_C)]	0.4751	Sí
F	[L,(F, D_C)]	0.4719	Sí

Se puede observar de la Tabla 4 que el movimiento de la tarea F genera la mayor disminución del costo, por esta razón se deja quieta y se continúa con la búsqueda de otro movimiento con la tarea restante que pueda generar un menor costo.

Nueva partición de inicio: [L,(F,D_C)]

Costo de la partición inicial: 0.4719

En la tercera iteración tenemos:

Tabla 5. Resultados de la tercera iteración del algoritmo Fiduccia-Mattheyses aplicado al diseño del nodo medidor.

Tarea movida	Partición resultante	Costo de la partición resultante	¿Se mejora el costo?
L	[ϕ,(L,F,D_C)]	0.4018	Sí

Después de realizar más iteraciones generando nuevas particiones no se obtuvieron mejores resultados que el presentado por la partición de la Tabla 5.

De los resultados obtenidos se concluyó que la mejor asignación corresponde a la implementación en hardware de todos los componentes que constituyen el nodo medidor.

3.3.5 Resultados experimentales

La validación total del sistema se cumplió utilizando la herramienta *CPN Tools,* con la que se logró realizar un análisis dinámico del sistema que permitió observar el efecto producido por la red de comunicaciones y la ejecución de los algoritmos en hardware y software, incluyendo la ejecución del planificador de tareas, que en este diseño tiene un tiempo de cómputo similar al de las tareas de la aplicación, razón por la cual la aplicación de las ecuaciones de análisis de planificabilidad a través de los métodos de porcentaje de utilización y tiempo de cómputo desprecia la ejecución del sistema operativo y no arrojaría un buen análisis.

La Figura 12 muestra los resultados obtenidos experimentalmente; en ella se puede apreciar la correspondencia entre los resultados experimentales y los de simulación. Las oscilaciones observadas se atribuyen principalmente al ruido presente en la medición de la velocidad a través de un tacogenerador y a los errores de redondeo debidos a la arquitectura seleccionada.

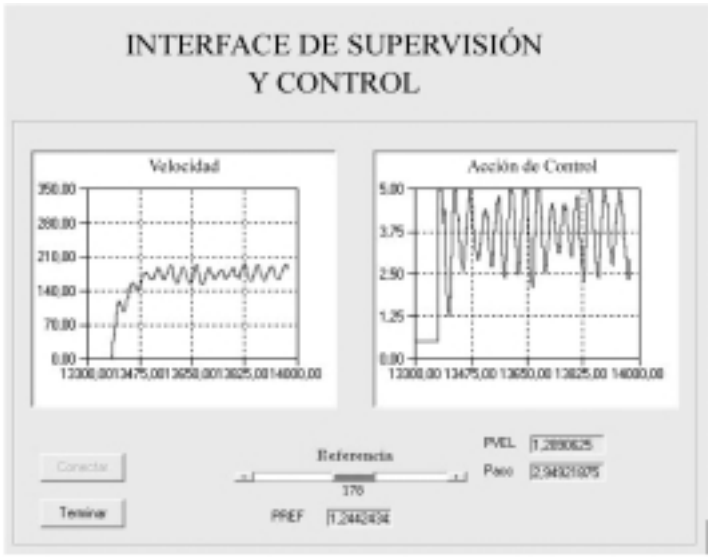


Figura 12. Resultados experimentales del sistema de control de velocidad.

4. Conclusiones y trabajo futuro

En este trabajo se presenta el diseño conjunto del algoritmo de control y de la plataforma de implementación de un sistema de control en red que regula la velocidad de un motor DC, para lo cual se utilizó un procedimiento de diseño de sistemas de control distribuido de tiempo real, que integra varias técnicas y metodologías para especificar y diseñar estos sistemas, permitiendo utilizar las ventajas presentes en las mismas.

Los resultados obtenidos permiten ver una buena correspondencia entre los análisis teóricos y los resultados experimentales, esto se atribuye a las consideraciones realizadas en el momento de calcular los parámetros del algoritmo de control y a los análisis realizados durante el diseño de la plataforma de implementación, lo cual permitió el cumplimiento de las restricciones del sistema.

En este diseño, para el desarrollo del módulo medidor fue posible integrar aspectos relacionados con la técnica de codiseño hardware-software.

El trabajo futuro será orientado a:

- Modelar nuevos componentes para el diseño arquitectural. Así como también, mejorar la notación de los mismos con el fin de considerar algunos aspectos relacionados con sistemas de diagnóstico, detección y corrección de fallos.
- Analizar y desarrollar nuevas plataformas de implementación para mejorar el desempeño de estos sistemas, y verificar su incidencia sobre el desempeño de los algoritmos de control.
- Modelar las interfaces mecánicas del sistema e incluir estos modelos dentro de los análisis del mismo.

5. Referencias

1. Martínez, D.; Velasco J., Pinedo, C. «Procedimiento de diseño de sistemas de control distribuido de tiempo real», Simposium Iberoamericano de Educación, Cibernética e Informática: SIECI 2004, Orlando, Florida-EE.UU, 21 al 25 de julio del 2004.
2. Burns, A., and Wellings, A. J. «HRT-HOOD a Structured Design Method for Hard Real-Time Systems». 1995.
3. Martínez, D.; Rodríguez, M. González, A. «Diseño de un Sistema de Monitoreo y Control de Tiempo Real Basado en Técnicas de Codiseño Hardware/Software», XVII Congreso Nacional y III Congreso Internacional de Informática y Computación de la ANIEI, Tepic, Nayarit, México, 20 al 22 de octubre de 2004.
4. Powel B., Real-Time UML. Second Edition, Addison-Wesley, Massachusetts, EE.U.U, 2000.
5. Martínez, D. «Procedimiento de diseño de sistemas de control distribuido de tiempo real». Tesis de Maestría en Automática. Universidad del Valle. Santiago de Cali, Colombia. 2004.
6. Walsh, G. C., Beldinan, O. L. Bushnell, «Asymptotic Behavior of Networked Control Systems», Submitted to Control Applications Conference, 1999.
7. Zhang, W. «Stability analysis of networked control systems», 2001.
8. Albertos, P. and Crespo, A. «Sistemas de Control en tiempo real: Avances en el diseño y realización», Corporación Universitaria Autónoma de Occidente, 2000.
9. El-Khoury, J. and Törngren, M. «Towards a Toolset for Architectural Design of Distributed Real-ime Control Systems», Proc. of Real-Time Systems Symposium (RTSS), 3-6 December, London, 2001, pp. 267-276.
10. López, M. and López, J. C. «On the Hardware-Software Partitioning Problem: System Modeling and Partitioning Techniques», ACM Transactions on Design Automation of Electronic System, Vol 8, No. 3, 2003, pp 269-297.
11. Pop, P.; Eles, P.; Pop, T.; Peng, Z. «An Approach to Incremental Design of Distributed Embedded Systems».
12. Martínez, D.; Arévalo, F.; Giraldo, C.; González, A. «Diseño de un nodo medidor para un sistema de control en red que regula la velocidad de un motor DC, basado en técnicas de Codiseño Hardware-Software». IV Congreso Internacional, Electrónica y Tecnologías de avanzada. Marzo 15 al 18 de 2005, Universidad de Pamplona, Pamplona-Colombia.